Assignment 5: Data Visualization

Rebecca Marx

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics (ENV872L) on data wrangling.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Use the lesson as a guide. It contains code that can be modified to complete the assignment.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document. Space for your answers is provided in this document and is indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file. You will need to have the correct software installed to do this (see Software Installation Guide) Press the `Knit` button in the RStudio scripting panel. This will save the PDF output in your Assignments folder.
6. After Knitting, please submit the completed exercise (PDF file) to the dropbox in Sakai. Please add your last name into the file name (e.g., "Salk_A04_DataWrangling.pdf") prior to submission.

The completed exercise is due on Tuesday, 19 February, 2019 before class begins.

Set up your session

1. Set up your session. Upload the NTL-LTER processed data files for chemistry/physics for Peter and Paul Lakes (tidy and gathered), the USGS stream gauge dataset, and the EPA Ecotox dataset for Neonicotinoids.

2. Make sure R is reading dates as date format, not something else (hint: remember that dates were an issue for the USGS gauge data).

```
#1
setwd("C:/Users/rsmar/OneDrive/Documents/Spring 2019/RFolder/Environmental_Data_Analytics")
#install.packages("gridExtra")
library(RColorBrewer)
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(colormap)
library(gridExtra)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------- tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
#Load data

PeterPaul.chem.nutrients <- read.csv("./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Proc

PeterPaul.nutrients.gathered <- read.csv("./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Pr

EPAecotox <- read.csv("./Data/Raw/ECOTOX_Neonicotinoids_Mortality_raw.csv")

USGS.flow.data <- read.csv("./Data/Raw/USGS_Site02085000_Flow_Raw.csv")
class(USGS.flow.data)
```

```
## [1] "data.frame"
```

```
#2

USGS.flow.data$datetime <- as.Date(USGS.flow.data$datetime, format = "%m/%d/%y")

USGS.flow.data$datetime <- format(USGS.flow.data$datetime,"%y%m%d")

create.early.dates <- (function(d) {
  paste0(ifelse(d > 181231,"19","20"),d)
  })

USGS.flow.data$datetime  <- create.early.dates(USGS.flow.data$datetime)

USGS.flow.data$datetime <- as.Date(USGS.flow.data$datetime, format = "%Y%m%d" )

class(USGS.flow.data$datetime)
```

```
## [1] "Date"
```

```
PeterPaul.chem.nutrients$sampledate <- as.Date(PeterPaul.chem.nutrients$sampledate, format = "%Y-%m-%d"

PeterPaul.nutrients.gathered$sampledate <- as.Date(PeterPaul.nutrients.gathered$sampledate, format = "%
```

Define your theme

3. Build a theme and set it as your default theme.

```
#3
RMtheme <- theme_bw(base_size = 11) +
  theme(plot.title = element_text(size = 15, color = "black", hjust = .5),
    axis.text = element_text(color = "black"),
    legend.position = "bottom", legend.text = element_text(size = 11), legend.title = element_text(size
```

Create graphs

For numbers 4-7, create graphs that follow best practices for data visualization. To make your graphs "pretty," ensure your theme, color palettes, axes, and legends are edited to your liking.

Hint: a good way to build graphs is to make them ugly first and then create more code to make them pretty.

4. [NTL-LTER] Plot total phosphorus by phosphate, with separate aesthetics for Peter and Paul lakes. Add a line of best fit and color it black.

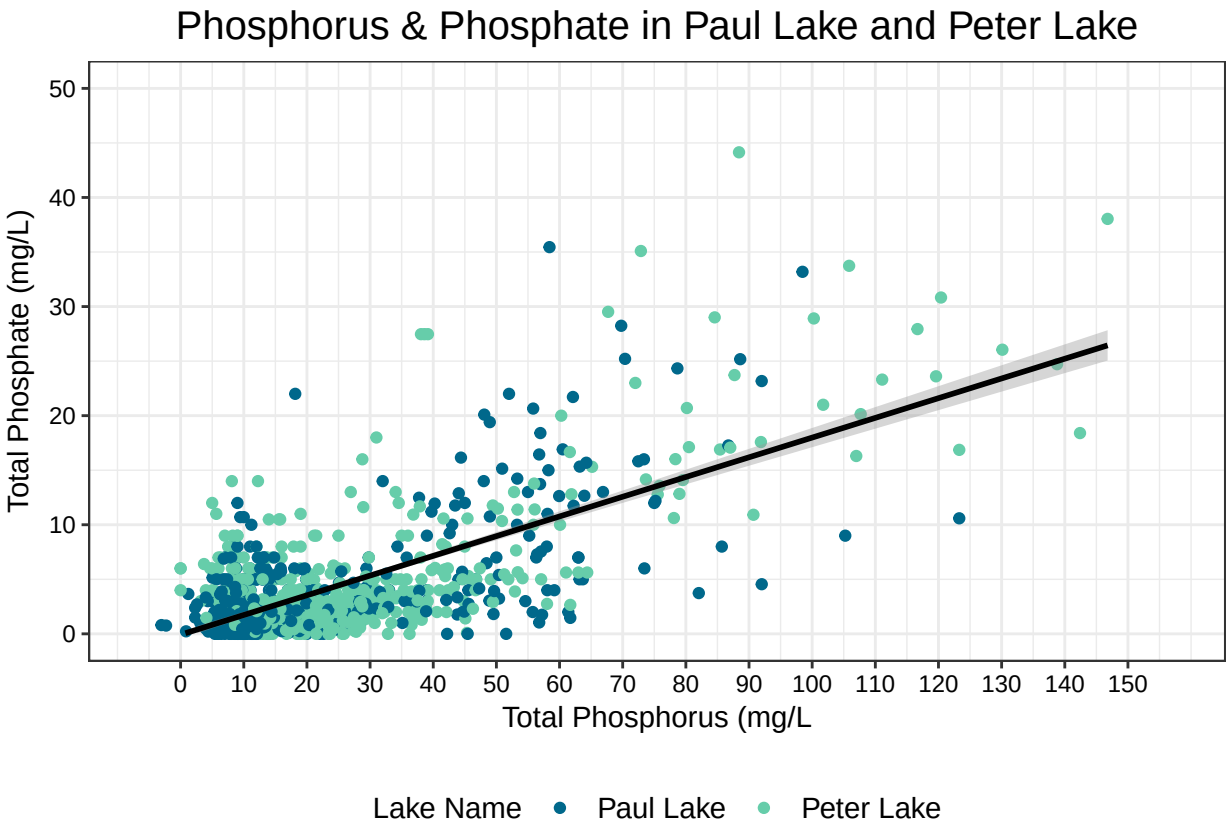
```
#4
legend_title_4 <- "Lake Name"
```

```
Q4graph <- ggplot(PeterPaul.chem.nutrients, aes(x = tp_ug, y = po4, color = lakename)) +
  geom_point() +
  ggtitle("Phosphorus & Phosphate in Paul Lake and Peter Lake") +
  ylab(expression("Total Phosphate (mg/L)")) +
  scale_y_continuous(limits = c(0,50)) +
  xlab(expression("Total Phosphorus (mg/L")) +
  scale_x_continuous(breaks=seq(0, 150, by=10)) +
  scale_color_manual(legend_title_4,values = c("deepskyblue4", "aquamarine3")) +
  geom_smooth(method = lm, color = "black") +
  RMtheme
print(Q4graph)
```

```
## Warning: Removed 22310 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 22310 rows containing missing values (geom_point).
```

```
## Warning: Removed 2 rows containing missing values (geom_smooth).
```




```
#
```

5. [NTL-LTER] Plot nutrients by date for Peter Lake, with separate colors for each depth. Facet your graph by the nutrient type.

```
#5

Peter.nutrients.gathered <- filter(PeterPaul.nutrients.gathered, lakename == "Peter Lake")

class(Peter.nutrients.gathered$sampledate)
```

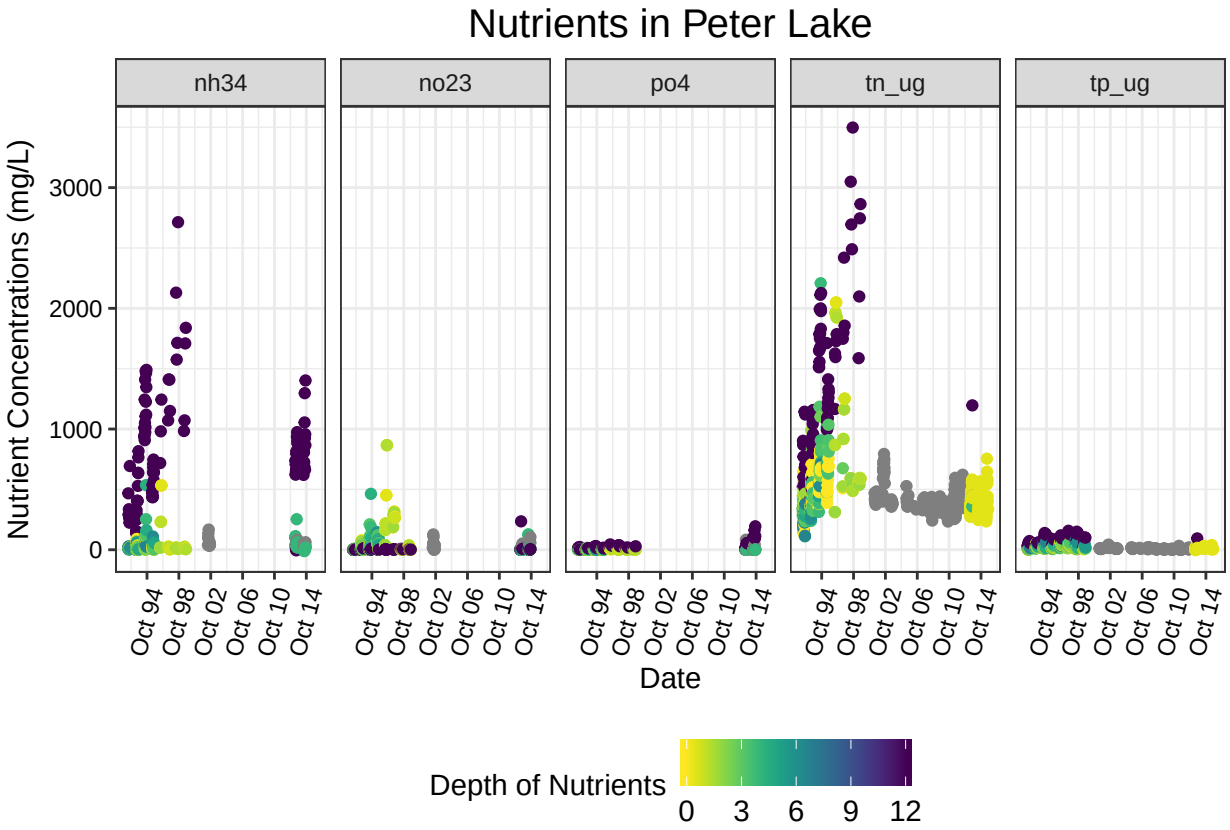
```
## [1] "Date"
```

```
Peter.nutrients.gathered$sampledate <- as.Date(Peter.nutrients.gathered$sampledate, format = "%Y-%m-%d")

class(Peter.nutrients.gathered$sampledate)
```

```
## [1] "Date"
```

```
legend_title_5 <- "Depth of Nutrients"

Q5_graph <-
  ggplot(Peter.nutrients.gathered) +
  geom_point(aes(x=sampledate, y = concentration, color = depth)) +
  facet_grid(~ nutrient) +
  scale_color_viridis(legend_title_5, option = "viridis", direction = -1) +
  ggtitle("Nutrients in Peter Lake") +
  ylab(expression("Nutrient Concentrations (mg/L)")) +
  xlab(expression("Date")) +
  scale_x_date(limits = as.Date(c("1992-01-01", "2015-12-31")),
  date_breaks = "48 months", date_labels = "%b %y") +
  RMtheme +
  theme(axis.text.x = element_text(angle = 75, hjust = 1))
print(Q5_graph)
```

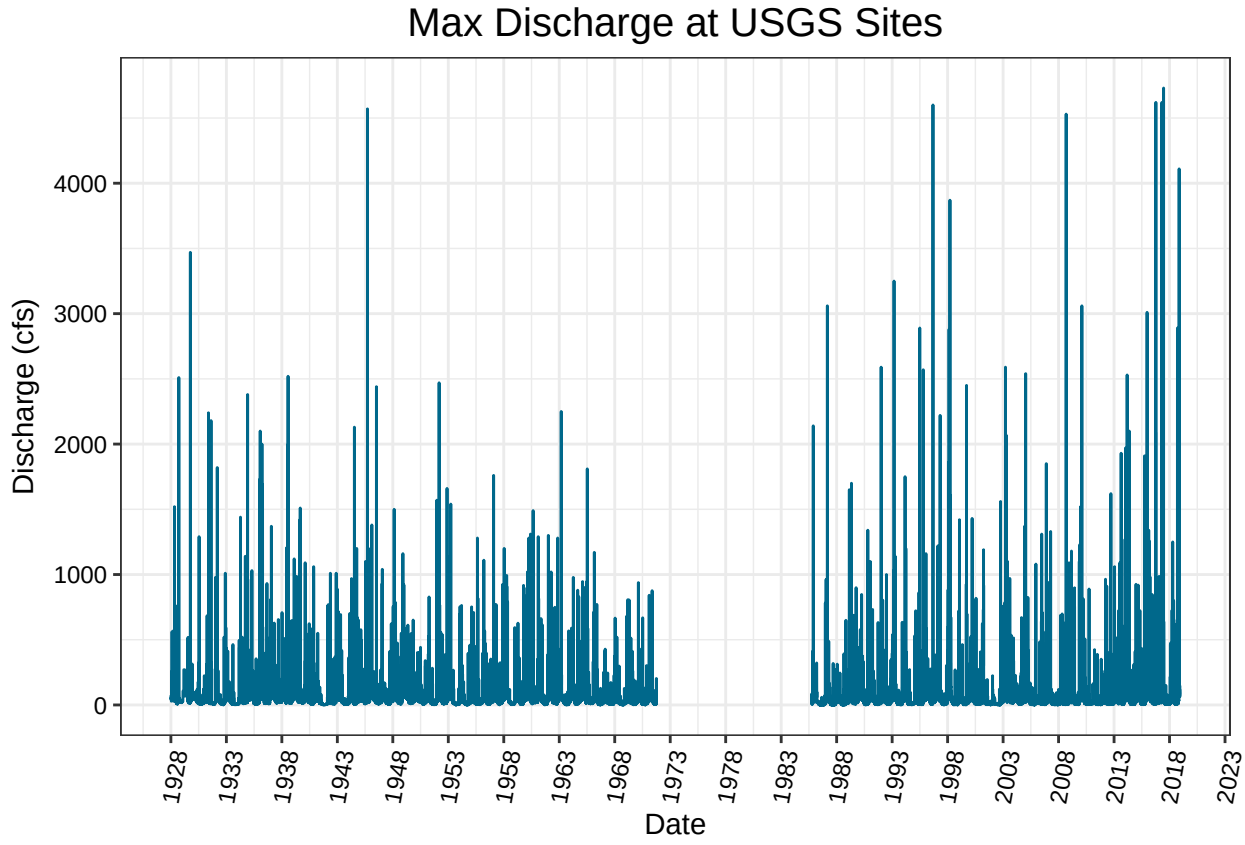
```
## Warning: Removed 503 rows containing missing values (geom_point).
```



6. [USGS gauge] Plot discharge by date. Create two plots, one with the points connected with geom_line and one with the points connected with geom_smooth (hint: do not use method = "lm"). Place these

graphs on the same plot (hint: ggarrange or something similar)

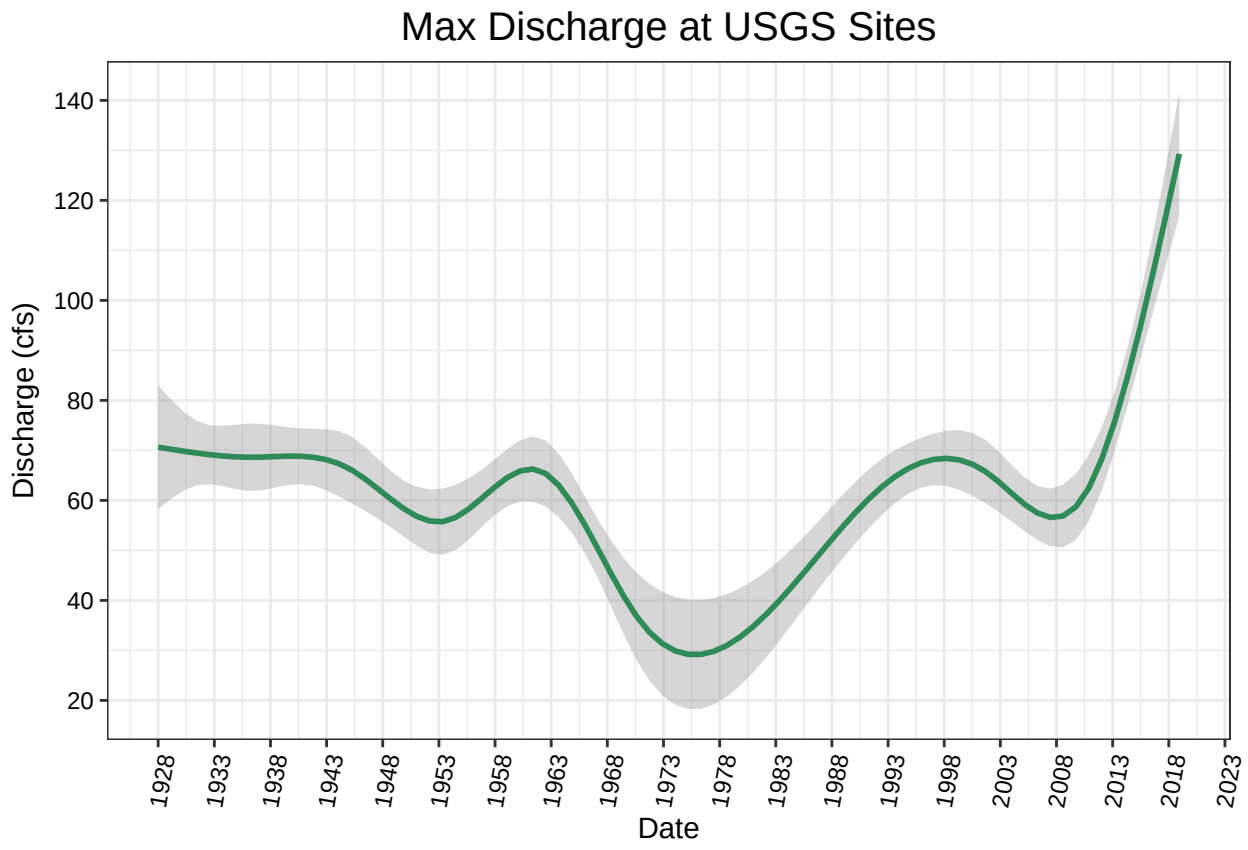
```
#6
#rename columns
colnames(USGS.flow.data) <- c("agency_cd", "site_no", "datetime", "discharge.max", "discharge.max.approv

Q6_graph.line <-
  ggplot(USGS.flow.data, aes(x=datetime, y = discharge.max)) +
  geom_line(color = "deepskyblue4") +
  ggtitle("Max Discharge at USGS Sites") +
  ylab(expression("Discharge (cfs)")) +
  xlab(expression("Date")) +
  scale_x_date(limits = as.Date(c("1928-01-01", "2018-12-09")),
  date_breaks = "5 years", date_labels = "%Y") +
  RMtheme +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
print(Q6_graph.line)
```




```
Q6_graph.smooth <-
 ggplot(USGS.flow.data, aes(x=datetime, y = discharge.max)) +
  geom_smooth(color = "seagreen") +
  ggtitle("Max Discharge at USGS Sites") +
  ylab(expression("Discharge (cfs)")) +
  scale_y_continuous(breaks=seq(0, 200, by=20)) +
  xlab(expression("Date")) +
  scale_x_date(limits = as.Date(c("1928-01-01", "2018-12-09")),
  date_breaks = "5 years", date_labels = "%Y") +
```

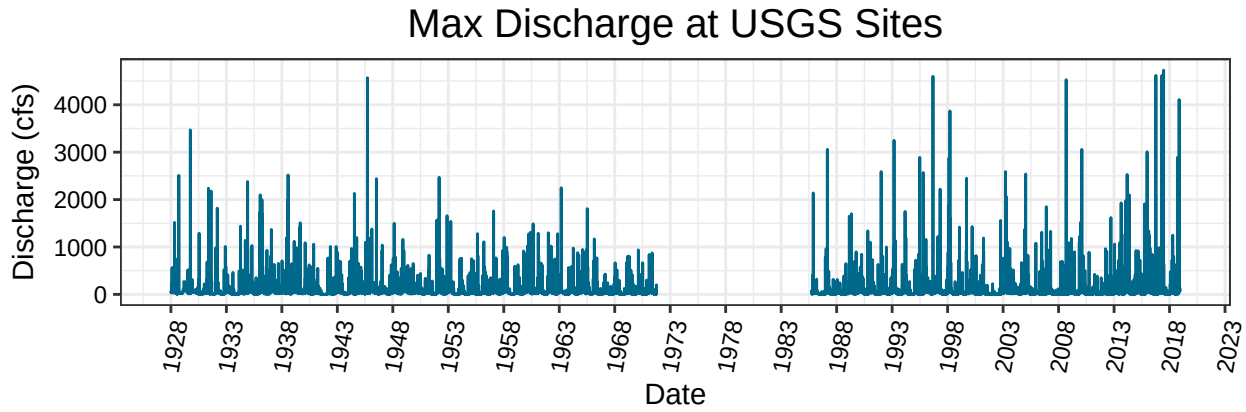
```
  RMtheme +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
print(Q6_graph.smooth)
```

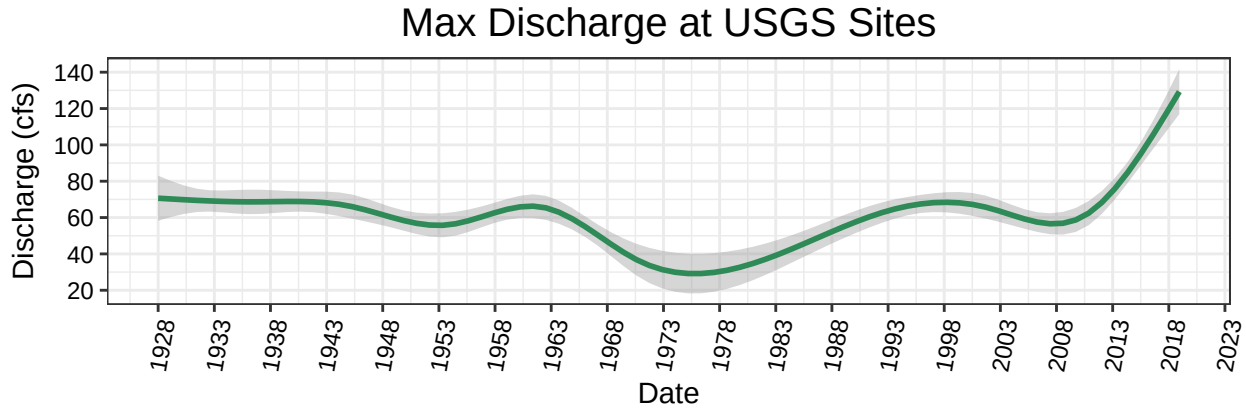
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 5113 rows containing non-finite values (stat_smooth).
```




```
Q6_graph_combined <- grid.arrange(Q6_graph.line, Q6_graph.smooth)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 5113 rows containing non-finite values (stat_smooth).
```






```
print(Q6_graph_combined)
```

```
## TableGrob (2 x 1) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (2-2,1-1) arrange gtable[layout]
```

Question: How do these two types of lines affect your interpretation of the data?

> Answer: The smooth chart is much easier to look at and determine the distribution of data given that it is less cluttered and includes the gray area indicating the standard deviation. In addition, geom_line displays a much larger range of discharge becuase it connects all of the outliers, whereas smooth averages the disharges. Smooth also interpolates what the dishcharge might be in the years where the value is "NA" (~1973 - 1985) whereas the line graph shows no values. If I did not know these were NAs, I might think discharge was quite low during that set of years by looking at the smooth graph rather than appreciating that there simply was no data for those years.

7. [ECOTOX Neonicotinoids] Plot the concentration, divided by chemical name. Choose a geom that accurately portrays the distribution of data points.

```
#7

legend_title_7 <- "Chemical Name"
Q7_graph <-
  ggplot(EPAecotox, aes(x = Chemical.Name, y = Conc..Mean..Std., color = Chemical.Name)) +
  geom_boxplot() +
  scale_color_manual(values = c("steelblue4", "firebrick3", "slateblue3", "tan2", "seagreen4", "plum4",
  scale_y_continuous(limits = c(0,2000)) +
```

```
  ggtitle("Chemical Concentrations") +
  ylab(expression("Concentrations (mg/L)")) +
  xlab(expression("Chemical")) +
  RMtheme +
  theme(axis.text.x = element_text(angle = 75, hjust = 1), legend.text = element_text(size = 8), legend
print(Q7_graph)
```

```
## Warning: Removed 33 rows containing non-finite values (stat_boxplot).
```